

## **REMARKS**

Reconsideration of the present application is respectfully requested. Each of the presently pending claims 1-33 have been amended, claims 1, 6, 9, 14, 19, 24, and 29 being independent.

In the Final Office Action mailed February 25, 2005, ("Final OA"), the Examiner restated the rejections set forth in the first office action. Specifically, the Examiner rejected claims 1-3, 6, 9-11, 14-16, 19-21, 24-26, and 29-31 under 35 U.S.C. § 102(e) as being anticipated by Chinnici et al., U.S. Patent Application Publication No. 2002/0188616. Claims 4, 7, 12, 17, 22, 27, and 32 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Chinnici in view of Steel, JR. et al., U.S. Patent Application Publication No. 2001/0056420. Finally, claims 5, 8, 13, 18, 23, 28, and 33 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Chinnici in view of Agesen, U.S. Patent No. 6,711,672.

### **Section 102 rejections**

The Final OA attempts to equate structured query language (SQL) with declarative languages, which is contrary to the teachings and definitions of the present application. Then this purported equivalence is employed to read the Chinnici reference on the pending claims. As set forth below, not only is the purported equivalence between SQL and a declarative language factually incorrect, the Final OA is attempting to use the Applicants' own disclosure to provide such a teaching, which would involve impermissible hindsight.

The above amendments are being presented to clarify that a simple SQL statement itself (as disclosed in Chinnici) does not read on declarative language function(s) or representation(s) as claimed. Specifically, at p. 11, lines 6-19, the application describes how, in an embodiment, SQL statements are "constructed from" or transformed into a functional language representation. It is true that once so transformed, the application states in this section that the query can be "treated as" functional language, however, the application is clear that a transformation is involved. As set forth

below, the transformation from a database query language, such as SQL, into declarative language functions is absolutely absent from the art of record, and it is improper to use Applicants' own teaching of such transformation in rejecting the present claims.

At page 3, paragraphs [0037] and [0038], the Chinnici reference merely describes conventional client-server database environments. It is understood that traditional two-tier and three-tier client-server database environments operate as described in Chinnici. As disclosed in paragraph [0037] of Chinnici, in a two-tier client-server environment, a database client connects directly to a database server, presenting for example a SQL query to the database server. Then the database server responds to the client with results, for example in a tabular data stream, or a stream of data logically formatted as a set of rows and columns. Paragraph [0038] merely describes a three-tier client-server environment, in which an application server is present. It is understood that, in a three-tier environment, a client communicates with an application server which then in turn connects to the database server. It is also understood that in any kind of a database environment that uses SQL, SQL statements are interpreted and corresponding data extracted from a database. Since Chinnici discloses nothing more than a conventional SQL parser in multi-tier environments, the Examiner appears to be taking the position that a conventional SQL parser anticipates the claims as presented. As set forth below, the present claims are distinct from a traditional SQL parser.

For example, several of the embodiments disclosed in the present application relate to querying data on small, lightweight devices such as Palm OS or Windows CE devices. One of the challenges identified in the present application is to provide a powerful query engine within the constraints of the disclosed exemplary lightweight devices. As set forth in the application, due to space constraints, a full-featured conventional SQL parser would not fit in a Palm OS or Windows CE device, and, therefore, any conventional SQL parser would have significant limitations when implemented in a lightweight environment. See pp. 20-21. Specifically, at pp. 20-21, the present application describes limitations inherent in traditional SQL interpreters when implemented on

devices with memory constraints such as the lightweight applications disclosed in the present application:

The traditional approach to the construction of query interpreters treats different concepts in the underlying query language in a non-uniform manner. The different concepts normally found are the operators found in the classical relational algebra

\* \* \*

This separation of concepts calls for a sophisticated interpreter infrastructure that honors this separation. . . . The resulting infrastructure tends to be complex. . . . Moreover, the separation of concepts results in a fixed (nested-loops based) skeleton of query execution which further restricts the set of possible SQL queries.

Application, p. 20, line 29 to p. 21, line 10.

As the application goes on to point out, no such restrictions exist in the method, system, and program product disclosed in the present application.

The novel systems and processes of the present application associate declarative language functions with query terms within database queries, so that, for example, in a lightweight device, a single, quite small, interpreter loop can cause the declarative language functions to be executed. See p. 14, line 26 to p. 15, line 4. Within the application, "declarative language" is defined on p. 8 as a "sets of definitions or equations describing relations which specify what is to be computed rather than how it is to be computed." According to this definition, SQL itself is not a "declarative language." The applications' teaching that a SQL query can be "constructed from functional language type functions" emphasizes this distinction. Accordingly, rather than equating SQL with a declarative language, the present application discloses transforming a SQL statement into a tree of declarative language functions. Nothing in the art of record teaches or suggests transforming SQL into declarative language functions.

In short, SQL is not a declarative language. Nothing in the art of reference teaches or suggests representing a database query in an intermediate declarative language format to provide an efficient, yet full-featured, database query language interpreter.

With respect to specific claim language, independent claims 1, 14, and 24 are patentable at least for the reason that none of the art of record either teaches or suggests "interpreting the queries by associating at least one declarative language function with the query terms" as recited in the claims. As to claim 6, none of the art of record teaches or suggests "converting the query language to a plurality of imperative language statements that represent a declarative language representation of the queries." Claims 9, 19, and 29, recite "representing the queries in accordance with a declarative language paradigm as a plurality of declarative language functions," which is neither taught nor suggested by any reference of record in the application.

Dependent claims recite further claim elements neither taught nor suggested by the art of record. For example, claims 2, 15, and 25 recite "converting the query language to an intermediate tree representation corresponding to the at least one declarative language function associated with the plurality of query terms, and thereafter converting the query to at least one data structure that is interpreted by an imperative language interpreter core to perform the queries," which is neither taught nor suggested by any reference of record. Claims 10, 20, and 30 recite a similar claim element and are similarly patentable for at least these reasons. Claims 3, 11, 16, 21, 26, and 31 further depend from the above dependent claims and are further patentable at least for the reason that none of the art of record teaches or suggests that the claimed declarative language functions be identified by a pointer to further code such that the declarative language function is treated as data within the plurality of imperative language statements.

#### Section 103 rejections

In addition to the section 102 rejections, the Examiner rejects the dependent claims that recite specific embodiments of declarative languages (claims 4, 7, 12, 17, 22, 27, and 32) in view of a generalized reference (Steele Jr) that mentions several known declarative languages. As set forth above, Chinnici fails to teach or suggest several elements of the present claims. Steele Jr does not remedy the deficiencies of Chinnici. Moreover, Applicants traverse these rejections for

lack of any motivation to combine. The purported motivation is “to enable system uses declarative language . . . due to programming conventions for converting database record to correspond to the query request as modified by the client computer system.” As understood, the purported motivation is found only in the teachings of the present invention. Only the present invention teaches that the use of declarative language functions to represent a database query permits a highly optimized and efficient query interpreter to operate in a very lightweight environment. The Examiner has presented no evidence of such motivation other than the disclosure of the application itself. Therefore, in addition to the deficiencies of Chinnici, the purported motivation to combine relies on impermissible hindsight.

Claims 5, 8, 13, 18, 23, and 33 are similarly rejected under section 103 based on a generalized reference (Agesen) that mentions specific imperative languages. These claims are patentable at least for the reasons set forth above for the deficiencies of Chinnici.

Conclusion

For the above stated reasons, Applicants respectfully suggest that none of the references of record, when considered singly or in combination, show or suggest the elements recited in the claims. Accordingly, Applicants respectfully request that a timely Notice of Allowance be issued in this case.

Should the Examiner have any questions, please contact the undersigned at (800) 445-3460. While the undersigned does not believe any additional fees are due in connection with this application, the Commissioner is hereby authorized to charge any additional fees associated with this communication or credit any overpayment to Deposit Account No. 09-0460.

Respectfully submitted,

HOVEY WILLIAMS LLP

By:



Jason E. Gordon, Reg No. 46,734  
HOVEY WILLIAMS LLP  
2405 Grand Boulevard, Suite 400  
Kansas City, Missouri 64108  
(816) 474-9050

ATTORNEYS FOR APPLICANTS